An Algorithmic Approach for Analyzing Wireless Networks with Retrials and Heterogeneous Servers

Nawel Gharbi
Computer Science Department
University of Sciences and Technology, USTHB
Algiers, Algeria
Email: ngharbi@wissal.dz

Leila Charabi
Computer Science Department
University of Sciences and Technology, USTHB
Algiers, Algeria
Email: leila.charabi@gmail.com

***Abstract*—Models with retrial phenomenon and heterogeneous servers arise in various wireless networks. This paper aims at presenting an approach for modeling and analyzing finite-source wireless networks with retrial phenomenon and heterogeneous servers using the Generalized Stochastic Petri Nets. This high-level formalism allows a simple representation of complex systems. Moreover, from the GSPN model, a Continuous Time Markov Chain can be automatically derived for the performance analysis. However, for important retrial networks, generating the Markov chain from the GSPN and solving it, require large storage space and long execution time. Hence, using the GSPN model as a support, we propose an algorithm for directly computing the infinitesimal generator of the GSPN model without generating neither the reachability graph nor the underlying Markov chain. In addition, we develop the formulas of the main stationary performance indices, as a function of the network parameters, the stationary probabilities and independently of the reachability set markings. Through numerical examples, we discuss the effect of the system parameters on performance.**

***Keywords*-Retrial phenomenon; Heterogeneous servers; Wireless networks; Generalized Stochastic Petri nets; Performance indices.**

I. Introduction

Models with retrial phenomenon are characterized by the feature that a customer finding all servers busy or unavailable, is obliged to leave the service area, but he repeats his request after some random period of time. These models play an important role in wireless cellular networks [4], [11], [12]. Significant references reveal the non-negligible impact of repeated calls, which arise due to a blocking in a system with limited capacity resources or are due to impatience of customers. For a systematic account of the fundamental methods and results on this topic, we refer the readers to [2], [3], [7].

Most studies on retrial models with finite source (population), assume that the service station consists of homogeneous (identical) servers. However, retrial models with heterogeneous servers arise in various practical areas as telecommunications and cellular mobile networks. In fact, these models are far more difficult for mathematical analysis than models with homogeneous servers, and explicit results are available only in few special cases and almost all studies are investigated only by means of queueing theory. In fact, we have found in the literature, only the few papers of Efrosinin [6] and Sztrik [10], [9] where heterogeneous servers case was considered using retrial queueing model, and the paper [8] where we have proposed the modeling and the analysis of multiclass retrial systems by means of colored generalized stochastic Petri nets.

The objective of this paper is to present an approach for modeling and analyzing performances of finite-source retrial networks with heterogeneous servers using the Generalized Stochastic Petri Nets (GSPNs) [1], [5]. From a modeling point of view, and compared to retrial queueing models, this high-level graphical formalism allows an easier description of the behavior of complex retrial networks, and it has shown to be a very effective mathematical model, appropriate for modeling and analyzing performance of parallel systems exhibiting concurrency and synchronization. Moreover, from the GSPN model, a Continuous Time Markov Chain (CTMC) can be automatically derived for the performance analysis. However, generating the Markov chain from the GSPN and solving it, still requires large storage space and long execution time, since the state space increases exponentially as function of the customers source size and servers' number. So, for important retrial networks, the corresponding models may have a huge state space. Hence, using the GSPN model as a support, we propose in this paper, an algorithmic approach for directly computing the infinitesimal generator without generating the reachability graph nor the underlying Markov chain. In addition, we develop the formulas of the main stationary performance indices, as a function of the number of servers of each class, the size of the customers source, the stationary probabilities and independently of the reachability set markings.

The paper is organized as follows. In Section 2, we describe the basic model of finite-source retrial networks with heterogeneous servers. In Section 3, the basic notions of GSPNs are reviewed. Next, we present the GSPN model describing retrial networks with heterogeneous servers. In Section 4, the proposed stochastic analysis approach is detailed. The computational formulas for evaluating exact performance indices are derived in Section 5. Next, based on numerical examples, we validate the proposed approach and we discuss the effect of system parameters on the performability of the system. Finally, we give a conclusion.

II. The basic model

We consider retrial networks with finite source (population) of customers of size L and a service station that consists of heterogeneous servers. Each customer is either free, under service or in orbit at any time. The input stream of primary calls is the so called quasi-random input. The probability that any particular customer generates a primary request for service in any interval $(t, t+dt)$ is $\lambda dt + o(dt)$ as $dt \to 0$ if the customer is free at time t, and zero if the customer is being served or in orbit at time t.

The servers are partitioned in two classes: Class C_1 and Class C_2, that is the servers of a given class have the same parameters. Each class C_j $(1 \leq j \leq 2)$ contains S_j identical and parallel servers. There are two possible states for a server: idle or busy (on service). If there is an idle server at the moment a customer request arrives, then the service starts immediately. The customer becomes "*under service*" and the server becomes "*busy*". Service times are independent identically-distributed random variables, whose distribution is exponential with parameter μ_1 if a class C_1 server is selected and μ_2 for servers of class C_2.

Each customer request must be served by one and only one server. On the other hand, we consider, the *Random Server discipline*, which means that, the server to which a request is assigned is chosen randomly among all idle servers, whatever their class. After service completion, the customer becomes free, so it can generate new primary calls, and the server becomes idle again. Otherwise, if all servers of the two classes are busy at the arrival of a request, the customer joins the orbit and starts generation of a flow of repeated calls exponentially distributed with rate ν, until he finds one free server. We assume that all customers are persistent in the sense that they keep making retrials until they receive their requested service.

As usual, we assume that the arrival, service and inter-retrial times are mutually independent of each other.

III. GSPN model of retrial networks with heterogeneous servers

In this section, we present the Generalized Stochastic Petri Net model describing finite-source retrial networks with two servers classes and random server discipline. To this aim, we give the basic notions of Generalized stochastic Petri nets (GSPNs).

A GSPN [1], [5] is a directed graph that consists of two kinds of nodes, called places (drawn as circles) and transitions that are partitioned into two different classes: timed transitions with rates of negative exponential distribution (represented by means of white rectangles), which describe the execution of time consuming activities and immediate transitions (represented by black rectangles), which model logic activities as synchronization.

The system state is described by means of markings. A marking is a mapping from P to IN, which gives the number of tokens in each place after each transition firing. A transition is said to be enabled in a given marking, if and only if each of its normal input places contains at least as many tokens as the multiplicity of the connecting arc, and each of its inhibitor input places contains fewer tokens than the multiplicity of the corresponding inhibitor arc. Moreover, timed transitions can fire only after an exponentially distributed delay, while enabled immediate transitions have priority over timed transitions and fire in zero time.

The firing of an enabled transition creates a new marking (state) of the net. The set of all markings reachable from initial marking M_0 is called the *reachability set*. The *reachability graph* is the associated graph obtained by representing each marking by a vertex and placing a directed edge from vertex M_i to vertex M_j, if marking M_j can be obtained by the firing of some transition enabled in marking M_i.

This graph consists of *tangible markings* enabling only timed transitions and *vanishing markings* in which at least one immediate transition is enabled. Since the process spends zero time in the vanishing markings, they are eliminated from the reachability graph by merging them with their successor tangible markings [1]. This elimination results in a *tangible reachability graph*, which is isomorphic to a continuous time Markov chain (CTMC).

In the following, we present the GSPN model describing finite-source retrial systems with two servers classes and random server discipline. In this model depicted in Figure 1, place *Cus_Free* represents the free customers, *Orbit* contains the customers waiting for the service, *Ser_Idle1* and *Ser_Idle2* indicate respectively the number of free servers of class C_1 and class C_2, while *Cus_Serv1* and *Cus_Serv2* model the busy servers of both classes. The arrival of a primary call causes the firing of the transition *Arrival*, which firing rate is marking dependent and equals $\lambda.M(Cus\_Free)$ (*infinite service semantics*) which is represented by the symbol # placed next to transition, because all free customers are able of generating calls, independently of each other.

The place *Choice* is then marked. Following the marking of both *Ser_Idle1* and *Ser_Idle2*, we have the following scenarios:

- If both places are empty (no free server), the immediate transition *Go_Orbit* is enabled, and a token is deposited in the place *Orbit*, which means that the customer asking for service joins the orbit and becomes a source of a flow of repeated calls exponentially distributed with rate ν.

  The firing of the transition *Retrial* corresponds to the generation of a repeated call from a customer in orbit. This transition has infinite servers semantics, since all customers in orbit can trigger repeated calls independently;

- If only one place (*Ser_Idle1* or *Ser_Idle2*) contains tokens and the other is empty, the immediate transition corresponding to the class with at least one free server (*Begin_Serv1* for the class C_1, and *Begin_Serv2* for class C_2) is enabled. Hence the customer starts its service and the server becomes busy.
- When both places are marked (i.e., each class contains at least one free server), the two immediate transitions *Begin_Serv1* and *Begin_Serv2*, which are already in *structural conflict*, come into *effective conflict*. As there is no priority order between the two classes C_1 and C_2, the same weight is assigned to both transitions as follows:

$$w(Begin\_Serv1) = w(Begin\_Serv2) = \omega$$

Thus, firing of one or other is probabilistic and the probability equals $1/2$. Hence, servers of both classes have the same chance of being selected to serve the customers requests.

By the end of service of a customer under a server of class C_1 (C_2 respectively), the timed transition *Serv_End1* (*Serv_End2* respectively) fires. As several servers may be busy at the same time, the semantics of these two transitions is *infinite servers*. After completion of service, the customer returns to state free (one token in place Cus_Free) and the server becomes available (one token is put in place *Ser_Idle1* or *Ser_Idle2*, according to the class to which the server belongs).

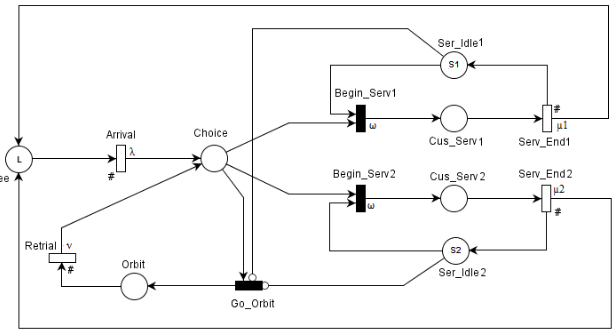

Figure 1. GSPN Model of finite-source retrial networks with two servers classes

IV. Stochastic analysis

Initially, the orbit is empty, all customers are free and all servers are available. Thus the initial marking can be expressed in this form:

$$\begin{aligned} M_0 &= \{M(Cus\_Free), M(Choice), M(Orbit), \\ &\quad M(Ser\_Idle1), M(Cus\_Serv1), M(Ser\_Idle2), \\ &\quad M(Cus\_Serv2)\} \\ &= \{L, 0, 0, S_1, 0, S_2, 0\} \end{aligned}$$

Whatever the values of L, S_1 and S_2, the conservation of customers and servers of the two classes, gives the following equations:

$$\begin{cases} M(Ser\_Idle1) + M(Cus\_Serv1) = S_1 \\ M(Ser\_Idle2) + M(Cus\_Serv2) = S_2 \\ M(Cus\_Free) + M(Cus\_Serv1) \\ +M(Cus\_Serv2) + M(Orbit) = L \end{cases} \tag{1}$$

Observing these three equations, we note that the system state at steady-state can be described by means of three variables (i, j, k), which we called a *micro-state* where :

- i represents the number of customers being served by servers of class C_1 (in place Cus_Serv1);
- j represents the number of customers being served by servers of class C_2 (in place Cus_Serv2);
- and k is the number of customers in orbit (in place $Orbit$).

Hence, having the micro-state (i, j, k), the markings of all places can be obtained, since $M(Ser\_Idle1) = S_1 - i$, $M(Ser\_Idle2) = S_2 - j$ and $M(Cus\_Free) = L - (i + j + k)$.

On the other hand, applying (1), we can deduce:

$$\begin{cases} 0 \leq i \leq S_1 \\ 0 \leq j \leq S_2 \\ 0 \leq k \leq L - (S_1 + S_2) \end{cases} \tag{2}$$

The CTMC corresponding to the proposed GSPN contains n micro-states corresponding to the accessible tangible markings, where n equals $(S_1 + 1).(S_2 + 1)[(L + 1 - S]$ and $S = S_1 + S_2$.

Thus, the corresponding infinitesimal generator Q is a $n \times n$ matrix, which is defined by :

$$\begin{cases} Q[(i,j,k),(x,y,z)] = \theta[(i,j,k),(x,y,z)] \\ Q[(i,j,k),(i,j,k)] = -\sum_{(l,m,n) \neq (i,j,k)} \theta[(i,j,k),(l,m,n)] \end{cases}$$

where $\theta[(i,j,k),(x,y,z)]$ is the transition rate from state (i,j,k) to state (x,y,z).

By analyzing the micro-states and the transitions rates of the CTMC, we obtain the following rates:

- $[0 \leq i \leq S_1 - 1, 0 \leq j \leq S_2 - 1]$: $(i,j,k) \xrightarrow{\frac{1}{2}(L-i-j-k)\lambda} (i+1,j,k)$ and $(i,j,k) \xrightarrow{\frac{1}{2}(L-i-j-k)\lambda} (i,j+1,k)$

- $[0 \le i \le S_1{-}1] : (i, S_2, k) \xrightarrow{(L-i-S_2-k)\lambda} (i{+}1, S_2, k)$,
- $[0 \le j \le S_2 - 1] : (S_1, j, k) \xrightarrow{(L-S_1-j-k)\lambda} (S_1, j + 1, k)$,
- $[0 \le k < L - (S_1 + S_2)] : (S_1, S_2, k) \xrightarrow{(L-S-k)\lambda} (S_1, S_2, k + 1)$,
- $[i > 0] : (i, j, k) \xrightarrow{i\mu_1} (i - 1, j, k)$,
- $[j > 0] : (i, j, k) \xrightarrow{j\mu_2} (i, j - 1, k)$,
- $[0 \le i \le S_1 - 1, 0 \le j \le S_2 - 1, k > 0] : (i, j, k) \xrightarrow{\frac{1}{2}k\nu} (i + 1, j, k - 1)$ and $(i, j, k) \xrightarrow{\frac{1}{2}k\nu} (i, j + 1, k - 1)$,
- $[0 \le i \le S_1{-}1, k > 0] : (i, S_2, k) \xrightarrow{k\nu} (i{+}1, S_2, k{-}1)$,
- $[0 \le j \le S_2 - 1, k > 0] : (S_1, j, k) \xrightarrow{k\nu} (S_1, j + 1, k - 1)$,

As a consequence, the infinitesimal generator can be automatically calculated by means of Algorithm 1 given below.

V. Performance measures

The aim of this section is to derive the formulas of the most important stationary performance. As the proposed models are bounded and the initial marking is a home state, the underlying process is ergodic. Hence, the steady-state solution exists and is unique. The infinitesimal generator Q corresponding to the GSPN model can be obtained automatically by applying the above algorithm. Then, the steady-state probability vector π can be computed by solving the linear system of equations:

$$\begin{cases} \pi.Q = 0 \\ \sum_i \pi_i = 1 \end{cases} \tag{3}$$

Where π_i denotes the steady-state probability that the process is in state M_i.

Having the probability distribution π, we can derive several exact stationary performance measures of finite-source retrial networks with two classes of servers applying the formulas given below. In following, $M_i(p)$ indicates the number of tokens in place p in marking M_i, A is the set of reachable tangible markings, and $A(t)$ is the set of tangible markings reachable by transition t.

- `Mean number of customers in the orbit:` This corresponds to the mean number of tokens in *Orbit*,

$$n_{Orb} = \sum_{i:M_i \in A} M_i(Orbit).\pi_i = \sum_{k=0}^{L-S} \sum_{i=0}^{S_1} \sum_{j=0}^{S_2} k.\pi_{i,j,k}$$

- `Mean number of busy servers of class` C_1`:` Note that this is also the mean number of customers under service by Class C_1, it corresponds to the mean number of tokens in place *Cus_Serv1*,

$$n_{busyC_1} = \sum_{i:M_i \in A} M_i(Cus\_Serv1).\pi_i = \sum_{k=0}^{L-S} \sum_{i=0}^{S_1} \sum_{j=0}^{S_2} i.\pi_{i,j,k}$$

Algorithm 1 Infinitesimal Generator Construction

```
for $k \leftarrow 0, L-S$ do
    for $i \leftarrow 0, S1-1$ do
        for $j \leftarrow 0, S2-1$ do
            $Q[(i,j,k),(i{+}1,j,k)] \leftarrow 1/2(L{-}i{-}j{-}k)\lambda$
            $Q[(i,j,k),(i,j{+}1,k)] \leftarrow 1/2(L{-}i{-}j{-}k)\lambda$
            $Q[(S1,j,k),(S1,j+1,k)] \leftarrow (L-S1-j-k)\lambda$
        end for
        $Q[(i,S2,k),(i+1,S2,k)] \leftarrow (L-i-S2-k)\lambda$
    end for
end for
for $k \leftarrow 0, L-S-1$ do
    $Q[(S1,S2,k),(S1,S2,k+1)] \leftarrow (L-S-k)\lambda$
end for
for $k \leftarrow 0, L-S$ do
    for $i \leftarrow 1, S1$ do
        for $j \leftarrow 0, S2$ do
            $Q[(i,j,k),(i-1,j,k)] \leftarrow i\mu_1$
        end for
    end for
    for $j \leftarrow 1, S2$ do
        for $i \leftarrow 0, S1$ do
            $Q[(i,j,k),(i,j-1,k)] \leftarrow j\mu_2$
        end for
    end for
end for
for $k \leftarrow 1, L-S$ do
    for $i \leftarrow 0, S1-1$ do
        for $j \leftarrow 0, S2-1$ do
            $Q[(i,j,k),(i+1,j,k-1)] \leftarrow 1/2.k\nu$
            $Q[(i,j,k),(i,j+1,k-1)] \leftarrow 1/2.k\nu$
        end for
        $Q[(i,S2,k),(i+1,S2,k-1)] \leftarrow k\nu$
    end for
    for $j \leftarrow 0, S2-1$ do
        $Q[(S1,j,k),(S1,j+1,k)] \leftarrow k\nu$
    end for
end for
```

- `Mean number of busy servers of class` C_2`:`

$$n_{busyC_2} = \sum_{i:M_i \in A} M_i(Cus\_Serv2).\pi_i = \sum_{k=0}^{L-S} \sum_{i=0}^{S_1} \sum_{j=0}^{S_2} j.\pi_{i,j,k}$$

- `Mean number of busy servers:` It corresponds to the sum of busy servers in both of the 2 classes

$$n_{busy} = n_{busyC_1} + n_{busyC_2} = \sum_{k=0}^{L-S} \sum_{i=0}^{S_1} \sum_{j=0}^{S_2} (i+j).\pi_{i,j,k}$$

- `Mean number of customers in the network`: Which is the total number of the mean number of customers in the orbit and those under service (by C_1 and C_2),

$$n = n_{Orb} + n_{busy} = \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0}^{S_2}(i+j+k).\pi_{i,j,k}$$

- `Mean number of free servers`: This corresponds to the sum of the mean number of free servers of both classes:

$$n_{Free} = S - n_{busy}$$

- `Effective customer arrival rate`: This represents the throughput of the transition *Arrival*,

$$\begin{aligned}\bar{\lambda} &= \sum_{i:M_i\in A(Arrival)} \lambda.M_i(Cus\_Free).\pi_i \\ &= \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0}^{S_2}\lambda.(L-i-j-k).\pi_{i,j,k} \\ &= \lambda.n_{CusFree}\end{aligned}$$

- `Effective customer retrial rate`: It corresponds to the throughput of *Retrial* transition,

$$\begin{aligned}\bar{\nu} &= \sum_{i:M_i\in A(Retrial)} \nu.M_i(Orbit).\pi_i \\ &= \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0}^{S_2}\nu.k.\pi_{i,j,k} = \nu.n_{Orb}\end{aligned}$$

- `Mean service rate of class` C_1: This corresponds to the throughput of the transition *Serv_End1*,

$$\begin{aligned}\bar{\mu_1} &= \sum_{i:M_i\in A(Serv\_End1)} \mu_1.M_i(Cus\_Serv1).\pi_i \\ &= \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0}^{S_2}\mu_1.i.\pi_{i,j,k} = \mu_1.n_{busyC_1}\end{aligned}$$

- `Mean service rate of class` C_2: This corresponds to the throughput of *Serv_End2*,

$$\begin{aligned}\bar{\mu_2} &= \sum_{i:M_i\in A(Serv\_End2)} \mu_2.M_i(Cus\_Serv2).\pi_i \\ &= \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0}^{S_2}\mu_2.j.\pi_{i,j,k} = \mu_2.n_{busyC_2}\end{aligned}$$

- `Availability of` s `servers in the system (among both classes)`:

$$\begin{aligned}A_s &= \sum_{i:M_i(Ser\_Idle1)+M_i(Ser\_Idle2)\geq s} \pi_i \\ &= \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0,i+j\leq S-s}^{S_2}\pi_{i,j,k}\end{aligned}$$

	Homogeneous case	Two servers classes system
Number of servers	4	S_1= 1,S_2= 3
Size of source	20	20
Primary call generation rate	0.1	0.1
Service rate	1	μ_1= 1, μ_2= 1
retrial rate	1.2	1.2
Mean number of busy servers	1.800 748	C_1: 0.521 865 C_2: 1.278 883 Total : 1.800 748
Mean number of source in the orbit	0.191 771	0.191 771
Mean primary call generation rate	1.800 748	1.800 748
Mean waiting time	0.106 495	0.106 495

Table II
VALIDATION IN THE HOMOGENEOUS CASE

- `Utilization of at least` s `servers`:This corresponds to the probability that at least s servers among the two classes are busy:

$$\begin{aligned}U_s &= \sum_{i:M_i(Cus\_Serv1)+M_i(Cus\_Serv2)\geq s} \pi_i \\ &= \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0,i+j\geq s}^{S_2}\pi_{i,j,k}\end{aligned}$$

- `The mean waiting time`: It's the mean period between the arrival of the customer and its service beginning. Using the Little's formula, the mean waiting time is given by:

$$\bar{W} = \frac{n_{Orb}}{\bar{\lambda}}$$

- `The mean response time`:

$$\bar{R} = \frac{n}{\bar{\lambda}}$$

VI. VALIDATION AND NUMERICAL EXAMPLES

In order to test the feasibility of our approach, we developed a $C\#$ code to implement the above algorithm (1) and the performance indices formulas. Next, we tested it for a large number of examples. In particular, in the homogeneous case, the results were validated by the Pascal program given in [7]. From table (II), we can see that both models give exactly the same results up to the 6th decimal digit.

	L	S_1	S_2	λ	ν	μ_1	μ_2
Figure 2	50	4	2	0.5	x axis	8	2
Figure 3	30	x axis	4	2	1	6	1
Figure 4	30	4	x axis	2	1	6	1

Table I
INPUT SYSTEM PARAMETERS

In the following, we present sample numerical results to illustrate graphically the impact of different system parameters on the mean response time. The input parameters are collected in table I.

The figure (2) shows the sensitivity of the mean response time to the retrial generation rate. Indeed, the response time decreases with the intensity of the flow of repeated calls, particularly when the retrial intensity is low (between 0.01 and 0.3), beyond the value 0.3, the influence becomes less significant.

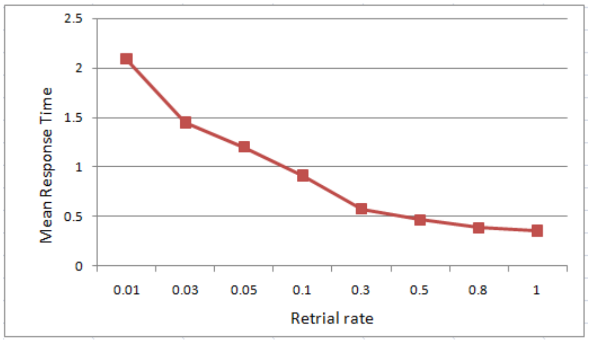

Figure 2. Mean response time versus retrial generation rate.

In figure 3, (4 respectively), we show the influence of the number of servers of C_1 (C_2 respectively) class, on the mean response time. We conclude that this last decreases with the increase of the servers number. However, the rate of influence of the number of servers in the C_1 class is faster than the influence due to increasing the number of servers of C_2 because the former is faster ($\mu_1 = 6$ vs $\mu_2 = 1$). In figure (3), the response time reached the optimum and stabilizes after a certain time (number of servers = 12). Hence, it is not interesting to invest in new servers in the C_1 class.

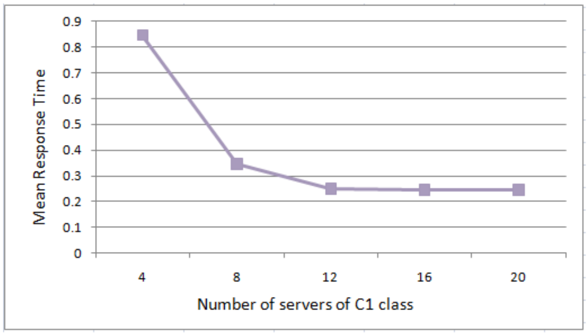

Figure 3. Mean response time versus C_1 class servers number.

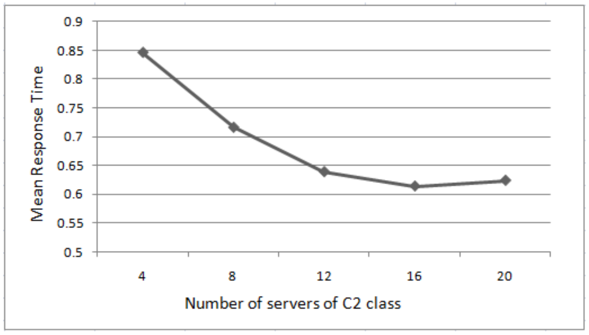

Figure 4. Mean response time versus C_2 class servers number.

REFERENCES

[1] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modelling with Generalized Stochastic Petri Nets*, John Wiley & Sons, New York, 1995.

[2] J.R. Artalejo and A. Gómez-Corral, *Retrial Queueing Systems: A Computational Approach*, Springer, Berlin, 2008.

[3] J.R. Artalejo, *Accessible bibliography on retrial queues: Progress in 2000-2009*, Mathematical and Computer Modelling, vol. 51, pp. 1071-1081, 2010.

[4] J.R. Artalejo and M.J. LOPEZ-HERRERO, *Cellular mobile networks with repeated calls operating in random environment*,Computers & operations research, vol. 37, no7, pp. 1158-1166, 2010.

[5] M. Diaz, *Les réseaux de Petri - Modèles Fondamentaux*, Paris, Hermès Science Publications, 2001.

[6] D. Efrosinin and L. Breuer, *Threshold policies for controlled retrial queues with heterogeneous servers*, Annals of Operations Research, vol. 141, pp. 139-162, 2006.

[7] G.I. Falin and J.G.C. Templeton, *Retrial Queues*, Chapman and Hall, London, 1997.

[8] N. Gharbi, C. Dutheillet, and M. Ioualalen, *Colored Stochastic Petri Nets for Modelling and Analysis of Multiclass Retrial Systems*, Mathematical and Computer Modelling, vol. 49, pp. 1436-1448, 2009.

[9] J. Roszik and J. Sztrik, *Performance analysis of finite-source retrial queues with nonreliable heterogeneous servers*, Journal of Mathematical Sciences, vol. 146, pp. 6033-6038, 2007.

[10] J. Sztrik, G. Bolch, H. de Meer, J. Roszik, and P. Wuechner, *Modeling finite-source retrial queueing systems with unreliable heterogeneous servers and different service policies using MOSEL*, Proc. of 14th Inter. Conf. on Analytical and Stochastic Modelling Techniques and Applications ASMTA'07, Pargue, Czech Republic, pp. 75-80, 2007.

[11] Tien Van Do, *A new computational algorithm for retrial queues to cellular mobile systems with guard channels*, Computers & Industrial Engineering, vol. 59, pp. 865-872, 2010.

[12] P. Tran-Gia and M. Mandjes, *Modeling of customer retrial phenomenon in cellular mobile networks*, IEEE Journal on Selected Areas in Communications, vol. 15, pp. 1406-1414, 1997.